SOFTWARE MAINTENANCE EXPERT SYSTEM (*SM*^{xpert})
A Decision Support Instrument

Alain April, Jean-Marc Desharnais, Ph.D.
École de Technologie Supérieure, 1100 Notre-Dame West, Montreal, Canada

Keywords: Knowledge-based system, software maintenance, maturity model, ontology, decision support.

Abstract: Maintaining and supporting the software of an organization is not an easy task, and software maintainers do not currently have access to tools to evaluate strategies for improving the specific activities of software maintenance. This article presents a knowledge-based system which helps in locating best practices in a software maintenance capability maturity model (***SM***^{mm}). The contributions of this paper are: 1) to instrument the maturity model with a support tool to aid software maintenance practitioners in locating specific best practices; and 2) to describe the knowledge-based approach and system overview used by the research team.

1 INTRODUCTION

Knowledge transfer of a large number of best practices, described in a maturity model, has proved difficult (Abran et al., 2004) . This is especially true during the training stage for an assessor or a new participant in a process improvement activity. It is also challenging to quickly refer to, or access, the right practice, or subset of practices, when trying to answer specific questions during or after a maturity assessment.

The maturity model ***SM***^{mm} contains a large number of concepts and information which are structured in many successive levels (April et al., 2004b, 2002, April et al., 2004a). The first is called the process domains level, and reflects the main process knowledge areas of a maturity model. In the ***SM***^{mm}, there are 4 process domains (*process management, maintenance request management, software evolution engineering* and *support to software engineering evolution*). Each process domain is broken down into one or more key process areas (KPAs). These KPAs logically group together items which conceptually belong together. A KPA is further divided into *roadmaps* with one or more *best practices*, spanning five ***SM***^{mm} maturity levels. The complete ***SM***^{mm} has 4 domains, 18 KPAs, 74 roadmaps and 443 best practices.

It would be beneficial to have a knowledge-based system (KBS) to help access this complex structure and large amount of information. A potential solution to this problem would be to develop a knowledge-based system for the ***SM***^{mm}. The proposed modeling of a software maintenance KBS is based on the van Heijst methodology (Van Heijst et al., 1997), which consists of constructing a task model, selecting or building an ontology (Uschold and Jasper, 2001), mapping the ontology onto the knowledge roles in the task model and instantiating the application ontology with this specific domain knowledge. According to van Heijst, there are at least five different types of knowledge to be taken into account when constructing such a system: tasks, problem-solving methods, inferences, the ontology and the domain knowledge[1]. For van Heijst, domain knowledge refers to a collection of statements about the domain (Van Heijst et al., 1997). The domain of this specific research is software maintenance, and it is divided into 4 process domains. Examples of statements are presented in section 3. At a high level, the ontology refers to a part of the software maintenance ontology proposed by (Kitchenham and et al., 1999) presented in section 4. The inferences, problem-solving methods and tasks are described at length in section 5. The tool environment and conclusion, as well as future work, are presented in sections 6 and 7. Section 2 begins by presenting the goals of the ***SM***^{mm} architecture.

[1] van Heijst uses the different types of knowledge in a more generic way than we do in this document, and these have been adapted for us by Desharnais, J.-M.,Application de la mesure fonctionnelle COSMIC-FFP: une approche cognitive,, UQAM,2004 Montréal

April A. and Desharnais J. (2005).
SOFTWARE MAINTENANCE EXPERT SYSTEM (SMxpert) - A Decision Support Instrument.
In *Proceedings of the Seventh International Conference on Enterprise Information Systems*, pages 142-148
DOI: 10.5220/0002526601420148

2 GOALS OF THE *SM*^{mm} ARCHITECTURE

The ***SM***^{mm} was designed as a customer-focused benchmark for either:

- Auditing the software maintenance capability of a service supplier or outsourcer; or
- Supporting the process improvement activities of software maintenance organizations.

To address the concerns specific to the maintainer, a distinct maintenance body of knowledge is required . The ***SM***^{mm} is also designed to complement the maturity model developed by the SEI at Carnegie Mellon University in Pittsburgh (CMMi, 2002) by focusing mainly on practices specific to software maintenance. The architecture of the model locates the most fundamental practices at a lower level of maturity, whereas the most advanced practices are located at a higher level of maturity. An organization will typically mature from the lower to the higher maturity level as it improves. Lower-level practices must be implemented and sustained for higher-level practices to be achieved.

3 *SM*^{mm} AND KNOWLEDGE STATEMENTS

Software maintainers experience a number of problems. These have been documented and an attempt made to rank them in order of importance. One of the first reported investigations was conducted by Lientz and Swanson (Lientz and Swanson, 1981). They identified six problems related to users of the applications, to managerial constraints and to the quality of software documentation. Other surveys have found that a large percentage of the software maintenance problems reported are related to the software product itself. This survey identified complex and old source code which was badly documented and structured in a complex way. More recent surveys conducted among attendees at successive software maintenance conferences (Dekleva, 1992) ranked perceived problems in the following order of importance (see Table 1). These are also examples of knowledge statements about the domain of software maintenance. Key to helping software maintainers would be to provide them with ways of resolving their problems by leading them to documented best practices.

Table 1: Top maintenance problems (Dekleva, 1992)

Rank	Maintenance problem
1	Managing fast-changing priorities
2	Inadequate testing techniques
3	Difficulty in measuring performance
4	Missing or incomplete software documentation
5	Adapting to rapid changes in user organizations
6	A large number of user requests in waiting
7	Difficulty in measuring/demonstrating the maintenance team's contribution
8	Low morale due to lack of recognition
9	Not many professionals in the field, especially experienced ones
10	Little methodology, few standards, procedures or tools specific to maintenance
11	Source code complex and unstructured
12	Integration, overlap and incompatibility of systems
13	Little training available to personnel
14	No strategic plans for maintenance
15	Difficulty in meeting user expectations
16	Lack of understanding and support from IT managers
17	Maintenance software running on obsolete systems and technologies
18	Little will for reengineering applications
19	Loss of expertise when employees leave

There is a growing number of sources where software maintainers can look for best practices, a major challenge being to encourage these sources to use the same terminology, process models and international standards. The practices used by maintainers need to show them how to meet their daily service goals. While these practices are most often described within their corresponding operational and support processes, and consist of numerous procedures, a very large number of problem-solving practices could be presented in a KBS which would answer their many questions about those problems. Examples are presented in section 5. When using the software maintenance ontology in the KBS, it was necessary to consider the structure of the maturity model relationship between the many process domains, roadmaps and practices. This problem is addressed next.

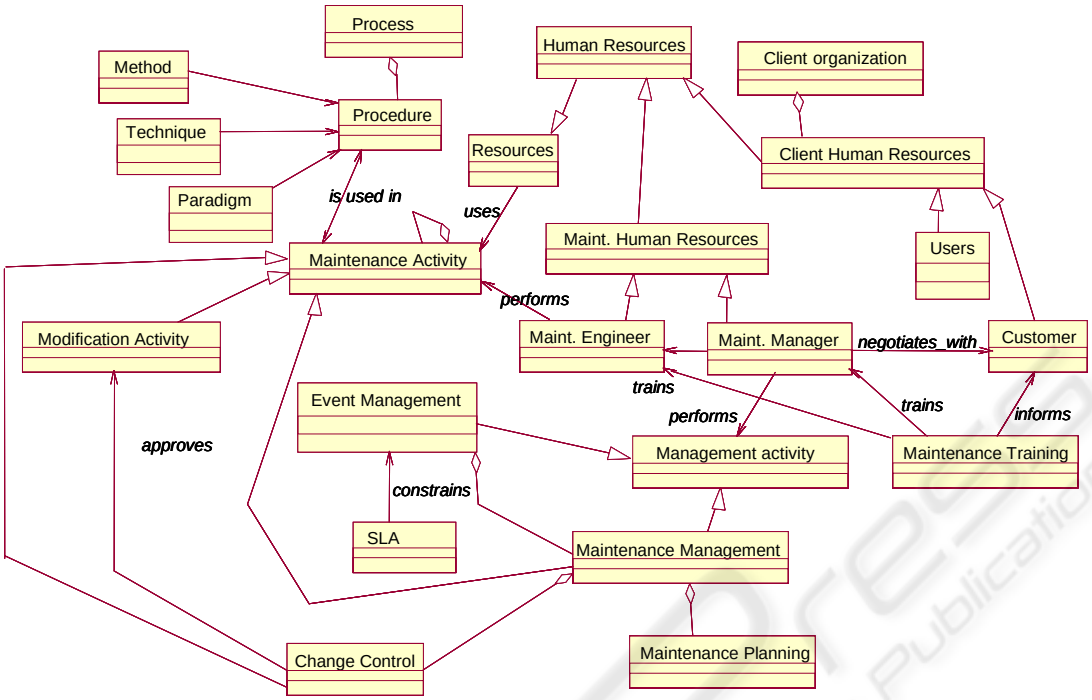


Figure 1: Part of the software maintenance ontology of (Kitchenham and et al., 1999)

4 ONTOLOGY OF THE SOFTWARE MAINTENANCE BODY OF KNOWLEDGE

We elected to implement only a subset of the ontology developed by Kitchenham et al. (1999) for the initial trial of this research project. The Kitchenam ontology was chosen because its author is well known in Software Engineering maintenance, The following authors also write on the subject (Vizcaíno et al., 2003), (Ruiz et al., 2004) and (Dias et al., 2003) from the point of view of the knowledge system. Figure 1 describes the different maintenance concepts considered surrounding a software maintenance activity. Software maintenance is highly event-driven, which means that some maintenance activities are unscheduled and can interrupt ongoing work. This subset of the ontology represents many, but not all, the concepts involved in responding to the questions related to the first problem identified by Dekleva: 'Managing fast-changing priorities.' Maintainers agree that this is the most important problem they face. How can they handle the fast-changing priorities of the customer? Solutions to this problem are likely to be found by using many paths through the maintenance concepts of the ontology. Navigation through these concepts should lead to associated concepts which are conceptually linked and likely to contribute to a solution, like the need for better event management, change control, maintenance planning, Service Level Agreements, maintenance manager negotiation, training, procedures, and so forth. Many more concepts must be involved to contribute to all aspects of the solution, but our purpose is to show the utility of a KBS in the software maintenance domain, and it therefore starts with a constrained number of concepts. Maturity models typically include the detailed best practices that could be of help in solving this type of problem. The main issue is that the best practice locations and their interrelationships are hidden in the layered architecture of the maturity model, specifically in its process domains, KPAs and roadmaps. It is therefore necessary to find a way to link this layered architecture with the maintenance concepts of the ontology and proceed to analyze the tasks required to build a KBS to support the maintainers in their quest for solutions. The next section describes the navigation concepts that have been implemented in ***SM**xpert*. The user of the KBS navigates using a sequence of tasks that will lead him through a further sequence of tasks.

5 TASK ANALYSIS

According to (Van Heijst et al., 1997), the first activity in the construction of a KBS is the definition of task analysis. Task analysis begins, at a high level, with a definition of an index of terms. This

index includes words commonly used in software engineering (see Figure 2). From this index, a subset of more restrictive words is identified. This subset is a list of keywords recognized specifically in software maintenance. Each keyword is then connected to one or more maintenance concepts. A maintenance concept, in software maintenance, is a concept found in the Software Maintenance Body of Knowledge and ontology (see Figure 2). Using the software maintenance ontology, every software maintenance problem identified by Dekleva has been linked to themes (questions) which help the user of the KBS to navigate to the part of the maturity model that will propose recommendations in the form of best practices.

Expanding the 5 high-level tasks in Figure 2, we propose 15 detailed tasks (see Table 2) which will help identify a best practice related to the SM^{mm}. The link between the maintenance concepts and the maturity model is made in the themes concept. Themes are questions which have been developed to hop from node to node in the ontology. A close look at Figure 1 reveals that the themes concept can send the user to another theme, to another maintenance concept (*up arrow*), or, finally, to a recommendation of the maturity model (*down arrow*). In Table 2, step 11, a number of themes, in the form of questions, are presented to the user to guide him through the network of maintenance concepts. For every best practice, there are a number of themes (or choices) from which the user can select (also called facts) which will lead to a specific recommendation. There are also a number of sub-tasks related to the maintenance processes and the maintenance best practices. (see Table 2). This step-by-step process corresponds to the establishment of a diagnosis on the basis of the identification of symptoms. It indicates probabilities of occurrence of a specific software maintenance problem. No symptom is sufficient by itself to confirm the existence of a specific problem. This is why we should use the word "diagnosis". The task model is used to help "diagnose" the current maintenance practice and map it to the maintenance model.

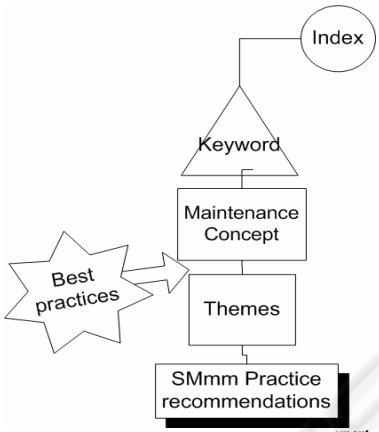
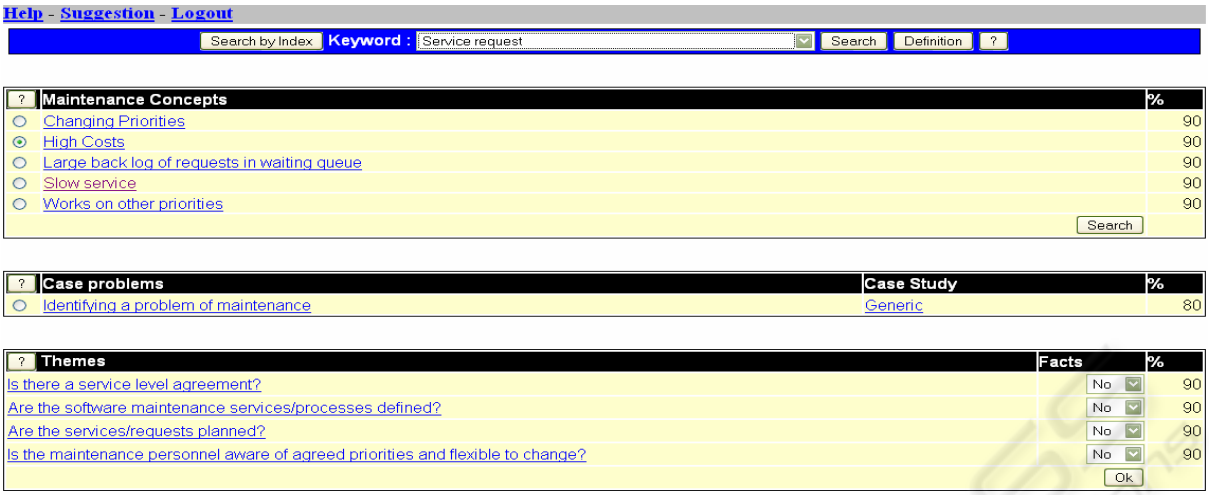


Figure 2: High-level view of SM^{xpert}

Appendix A shows how the KBS helps the user answer the following question: How do we accept or reject a new maintenance request?

6 TOOL ENVIRONMENT

The SM^{xpert} KBS was built using Java script and XML, and supports the SM^{mm}. The architecture, design and implementation details of this KBS are similar to those of the COSMIC KBS (Desharnais, 2003) which was developed as a diagnostic tool to help IT personnel in the estimation of functional size. The design of the KBS is based on using both the case-based and ruled-based approaches (Desharnais et al., 2002). The SM^{xpert} KBS was developed by two Master's degree students from the University of Namur, Belgium, during a research exchange program with our university (Desharnais et al., 2004). There is still a great deal of work required to populate the knowledge base for all the SM^{mm} practices to allow users to obtain answers to all the software maintenance problems identified by Dekleva. Figure 3 shows an example of the user layout. In this case, the user requests a recommendation in a case where the service request is very costly. A number of questions (themes) are asked by the system. According to the answers, there will be a specific recommendation which could either suggest further research or provide an opinion. There are also interfaces for both the administrator and the expert. The administrator interface manages access to SMXpert, while the expert interface gives the expert the option of adding new keywords, concepts, cases, themes and recommendations.



4. Answer the questions

Figure 3: *SM*^{xpert} user interface layout

7 CONCLUSION AND FUTURE WORK

Identifying the best practices in a maturity model is a difficult task, considering their number and the multiple appropriate answers associated with each of them. Our hypothesis is that a KBS could help in finding an appropriate recommendation. The next step in this research project is to populate the KBS, validate the results with experts in the domain and determine whether or not the KBS is a useful support tool for training on the content of the maturity model. It will be also necessary to improve the interface, mainly for the sake of the expert.

REFERENCES

Abran, A., Moore, J. W., Bourque, P., Dupuis, R. and Tripp, L.,*Guide for the Software Engineering Body of Knowledge (SWEBOK)*, Ironman version, IEEE Computer Society Press: Los Alamitos CA,2004; 6-1-6-15, Montréal, http://www.swebok.org [27 January 2005].

April, A., Abran, A. and Dumke, R. *SMCMM Model to Evaluate and Improve the Quality of the Software Maintenance Process: Improvements, traceability and conformity to standards*, CSMR 2004 8th European Conference on Software Maintenance and Reengineering, (2004a) Tampere (Finland)

April, A., Abran A. and Dumke, R. *Assessment of Software Maintenance Capability: A model and its Design Process*, IASTED 2004, Conference on Software Engineering (2004b), Innsbruck (Austria)

CMMi (Ed.) (2002) *Capability Maturity Model Integration for Software Engineering (CMMi), Version 1.1*, CMU/SEI-2002-TR-028, ESC-TR-2002-028, Carnegie Mellon University.

Dekleva, S. M. *Delphi Study of Software Maintenance Problems*, International Conference on Software Maintenance (CSM 1992) (1992) IEEE Computer Society Press: Los Alamitos CA

Desharnais, J.-M., *Application de la mesure fonctionnelle COSMIC-FFP: une approche cognitive*, UQAM, Montréal, 2003

Desharnais, J.-M., *Application de la mesure fonctionnelle COSMIC-FFP: une approche cognitive*, UQAM, Montréal, 2004

Desharnais, J.-M., Abran, A., Mayers, A., Buglione, L. and Bevo, V. *Knowledge Modeling for the Design of a KBS in the Functional Size Measurement Domain*, KES 2002, IOS Press, Crema, Italy

Desharnais, J. M., Abran, A., Mayers, A., Vilz, J. and Gruselin, F. (2004), *Verification and validation of a knowledge base system*, KI, Special Issue on Software Engineering for Knowledge-based Systems, Germany, **3**.

Dias, M. G., Anquetil, N. and Oliveira, K. M. (2003), *Organizing the Knowledge Used in Software Maintenance*, Journal of Universal Computer Science, **9, 7** 64-658.

Durkin, J. (1994) *Expert system: Design and Development*, Prentice Hall, New York.

Kitchenham, B. and et al. (1999), *Towards an Ontology of Software Maintenance*, J. Softw. Maint:Res. Parct., **11(6):**365-389.

Lientz, B. and Swanson, E. (1981), *Problems in Application Software Maintenance*, Communications of the ACM, **24, 11,** 763-769.

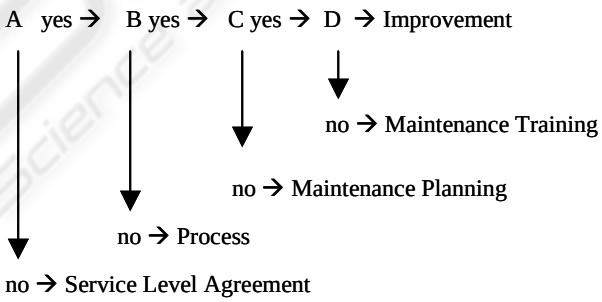
Ruiz, F., Vizcaino, A., Piattini, M. and Garcia, F. (2004) *International Journal of Software Engineering and Knowledge Engineering*, **14, 3** 323-349.

Uschold, M. and Jasper, R. (2001), *An ontology for the management of software maintenance projects*, In Industrial Knowledge Management: a micro-level approach,Bedford (UK), pp. 549-563.

Van Heijst, G., Schreiber, A. T. and Wielinga, A.,Using *Explicit Ontologies in KBS Development*, 2003 University of Amsterdam, Department of Social Science Informatics, Amsterdam, 1997

Vizcaíno, A., Favela, J. and Piattini, M. *A multi-agent system for knowledge management in software maintenance*, KES 2003 (2003), Springer Verlag, Oxford, UK

Appendix A: Task description of the KBS using Dekleva's first problem

NO.	TASK	EXAMPLE
1.	**Accessing the index**	The user enters a word that will identify a suggested keyword. As an example, the user enters: *Change in Priority*
2.	**Choosing a resulting keyword**	The user will enter a keyword that will help the KBS find the most closely related KPA and roadmap concepts. The system presents the following keywords: Change Management, Change Control, Staff Rotation, Event and Service Request, Service Level Agreement. The user chooses: *Event and Service Request*
3.	**Searching for a related software maintenance concept**	The KBS presents the maintenance concepts (which are related to the KPA and roadmap) to the user.
4.	**Giving priority to concepts**	The KBS will present the concepts in order of priority to the user. A percentage is related to each concept. The expert has previously established this percentage. As an example: *1) Event, 2) Process, 3) SLA, 4) Resource, 5) Change Control and 6) Maintenance Manager*.
5.	**Choosing a maintenance topological concept**	The user chooses one or multiple maintenance concepts, *Event* in our example
6.	**Displaying themes**	With *Event*, there are 5 themes presented to the user in the forum of questions: A) Is there a Service Level Agreement ? B) Are the software maintenance services/processes defined ? C) Are the services/requests planned ? D) Are the maintenance personnel aware of agreed priorities and amenable to change?
7.	**Choosing the status of each theme**	The user will find facts for each practice (theme). He can answer yes or no to any of the themes.
8.	**Rating the status (facts)**	An algorithm based on Bayesian Theory (Uschold and Jasper, 2001) is used to calculate the rate (MYCIN approach). The algorithm rates the facts chosen.
9.	**Displaying the results**	The resulting percentage relating to the best request management is shown to the user.
10.	**Assessing the results**	The formula is based on Bayesian Theory, as explained by (Durkin, 1994). Case 1 – CF(CP) = CF(Theme1) = q_choice_perc*P_Q_perc Case 2 – CF(CP) = CF(Theme1) *CF(Theme2) Case 3 – CF1(Theme) = CFcombine[CF(Theme1, CF(Theme2)] CF(CP) = CFcombine[CF1(Theme), CF(Theme3)] Etc.
11.	**Recommendation/explanation**	 The KBS will recommend the following solution (simplified for this paper):
12.	**Displaying other best practices**	Another part of the recommendation will show a different option, like: route request to account manager, interrupt work and insert in list of work, insert minor enhancement in list of work.
13.	**Displaying an explanation**	There is also the possibility of an explanation. In our case, the explanation takes up one page and could not be presented here due to lack of space (April et al., 2004b)
14.	**Acceptability**	Depending on the case that the user has to solve, the recommendation/explanation will be accepted or rejected. In our case, the user accepted the recommendation because it was not necessary to refer the change request to another group based on the criteria.
15.	**Choosing best practices (new)**	The process could start again. In this example, the user decided to stop because he considered he had enough information about the case. In a more complex situation, more choices could be necessary.